


UNIVERSITAT POLITÉCNICA DE CATALUNYA (UPC)
UNIVERSITAT DE BARCELONA (UB)
UNIVERSITAT ROVIRA I VIRGILI (URV)

Master in Artificial Intelligence

Master of Science Thesis

Using Deep Learning for Social Analysis in Egocentric Images

Khalid Mahyou

Supervisors:
Dr. Brais Cancela
Dr. Petia Radeva

October 2017

Abstract

In recent years, face recognition has achieved remarkable results in recognizing faces in images and videos. However, the task of cluster faces in unconstrained images and video frames is still an open problem, which generally requires time consuming steps. We explore in detail and propose a system to cluster faces from unconstrained images. This system can be divided mainly in two big steps: (i) given an input image, align the faces found in the image and pass it through a deep convolutional neural network (DCNN) to get a compact representation of the face, and (ii) cluster the face images by their feature representation. In the first step, we provide a detailed explanation of different feature extractor models as well as our fine-tuned model. In the second step several clustering algorithms were used and tested. Further, our objective was to apply this model to egocentric images to study people's social behaviour and demonstrate how to tackle this kind of images, which added several more challenges to the list. Egocentric images have less quality, are often blurred, noisy, have occlusions, and their view range is very limited. All these difficulties are specially important in face recognition and clustering problems, as it is necessary to see the whole face or at least an important part of it infer it representation. The achieved results from our system is compared to state of the art results, where we achieve 94% of F_1 score on LFW dataset and outperformed state of the art results in some IJB-B clustering protocols (i.e. 94% on IJB-B-128, 94% on IJB-B-256, 85% on IJB-B-512 and 80% on IJB-B-1024 F_1 score).

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The task of accurately identifying and grouping people has always been a very human process that we perform in our daily lives without even notice it or thinking about it. In the recent decade, the availability of powerful and low-cost computers, the development of high-performing computer models, as well as the vast amount of publicly available data, had a huge impact in the academia as well as in the industry that developed an enormous interest in automatic processing of digital media (i.e. digital images or videos) in a variety of applications including but not limited to, multimedia management, biometric authentication, surveillance, law enforcement, etc. This particular situation lead to an increase in research and development in automatic face verification (is this the same person), face recognition (who is this person) and clustering (find common people among these faces).

In the recent years, as the development of surveillance cameras and mobile devices continues to grow, so does the size of image and video collections. For instance, in the context of law enforcement (i.e. forensic investigations), this represents a major issue as the exploitation of such imagery must proceed in a timely manner. Other cases require the investigation of social media collections: including identifying victims in some criminal acts, an understanding of which persons exist in a collection of social media (such as, for instance, imagery from gang networks), organizing personal pictures (from hard drives or cloud storages) and summarizing imagery from social media.

In this work, we want to tackle the problem of given a large number images, to be able to find automatically all faces among the images and cluster them into the individual identities present in the data without knowing the exact number of clusters in advance nor the number of identities. Thus, the total number of different partitions can range from few

hundred to thousand and even to millions. This particular situation is encountered in a different number of application scenarios as stated in the previous paragraphs (i.e. ranging from social media to law enforcement).

Despite extensive studies on general clustering algorithms over the past few decades, face image clustering has received relatively little attention, although being an important role in some of the most critical applications. This is due because clustering of face images remains a difficult task. The difficulties are mainly three-fold: (i) since face images of a person may have variations in illumination, facial expressions, occlusion, age, and pose, it is challenging to measure the similarity between two face images, (ii) without knowing the actual number of clusters, many well-established clustering algorithms, such as k-means, may not be effective and (iii) a lack of understanding of what clustering algorithms are the most accurate given a large number of samples and/or subjects, and well tuned facial features.

Face clustering, or face discovery, algorithms provide meaningful partitions for given face image sets by combining faces with similar appearances while separating dissimilar ones. Ideally, face images in a partition should belong to the same identity, while images from different partitions should not. Clustering is also important when we need large amount of data to train a deep convolutional neural network (DCNN) [20] for face verification, classification, or object detection. For instance, due to diversity of different datasets and considering that very large dataset has been built automatically from the outputs of different search engines, labelling errors could adversely affect the training of such deep networks. An effective approach to avoid this sort of errors is to apply a reliable clustering algorithm on such training dataset to harvest sufficient number of images that can be used for training a DCNN.

One particular application we want to address in this work, is the case of clustering faces in egocentric images. Recently, the trend of people towards the use of wearable devices, and concretely wearable cameras to automatically record their daily live moments has considerably increased. Among all the available cameras, photo-cameras that take pictures at a lower frame-rate (i.e. 1 picture every 120 seconds), without needing to recharge for several consecutive days, are more suitable for long time acquisition. Images collected over a long period of time contain valuable information about the lifestyle of the user. Among different types of application using these kind of pictures, i.e. ranging from memorability perspective to medical applications, the most prominent ones are those that the user shows specific emotions, which without a doubt involve in social interactions. Due to this fact, the clustering of faces in egocentric images unveil less noticed matters about the social life of the user: for instance, with whom does she/he interacts the most?, how many times she/he

has been with her/his friends last month?, etc. and hence they are potentially beneficial for prevention of non-communicative diseases associated with unhealthy trends and risky profiles, such as obesity, depression or people suffering from mild cognitive impairment in elderly people [9] [40].

We perceive the following contributions related with this work: (i) the use of a well-known and simple, yet powerful, clustering algorithm to tackle the problem of face clustering from unconstrained images (still images and video frames), (ii) mid and large-scale face clustering experiments using unconstrained face datasets, namely LFW and IJB-B, with state of the art face representation learned for face recognition based on convolutional deep networks [28], (iii) a preliminary investigation of the applicability of the presented face clustering method to egocentric images (images that are taking by a wearable device) using EDUB-Obj dataset, and (iv) yet our approach seems quite straightforward, compared to more advanced methods such the one by Shi et al. [34], where they use very complex systems, such Res-nets and tackling the clustering problem as CRFs formulation, our method still very competitive and output results compared to state of the art and in some cases outperform them (see section 4.3)

1.1 Outline

This document is divided into five chapters, including this current introductory one.

Chapter 2 provides a self-contained overview of related work in face recognition and face clustering as well as the work done in egocentric images. Recent advances in both cases are reviewed and presented.

Chapter 3 provides a summary of the technical machine learning components needed for building a face clustering pipeline. Different concepts, including the extraction of faces from images, deep features using DCNNs and clustering are introduced.

Chapter 4 contains the evaluation results based on well known benchmarks datasets. An explanation of the datasets as well as the performance of the used algorithms are provided, including comparisons with related work.

Chapter 5 summarizes this work and gives some future ways of improvement and research interests.

Chapter 2

Related Work

The clustering problem, a powerful tool for data analysis, has been well studied in pattern recognition, statistics, and machine learning literature [14]. However, less studied is the challenging problem of clustering face images. An important consideration in clustering face images is that since there is no universally agreed upon face representation or distance metric, the clustering results depend not only on the choice of clustering algorithm, but also on the quality of the underlying face representation and metric.

2.1 Clustering algorithms

Clustering algorithms broadly can be categorized into partitional and hierarchical approaches. Both approaches build upon a similarity matrix or graph G(V, E) defined for the given dataset. For partitional approaches [22] [33] [23] [50], a simply a division of the set of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset. In the other hand, hierarchical approaches [18] create a set of nested clusters that are organized as a tree. This category can be further divided into two distinct types: (i) agglomerative (bottom -up): which starts with the points as individual clusters and, at each step, merge the closest pair of clusters until only singleton cluster remains, and (ii) divisive (top-down): which starts with one, all-inclusive cluster and, at each step, splits a cluster until only singleton clusters of individual points remain.

Gong et al. [10] develop a version of k-means clustering which is suitable for handling large datasets by encoding their feature vectors to binary vectors, and then using an indexing scheme to support constant time lookup of cluster centres for the assignment step

of k-means. They apply their binary k-means algorithm to a subset of the ImageNet dataset [30], containing 1.2 million general object images in 1,000 classes.

Liu et al. [10] in their work, followed next steps: (i) extract Haar wavelet features from images, then (ii) apply a distributed algorithm consisting of an approximate nearest neighbour step, (iii) generate an initial set of clusters by applying a distance threshold to the nearest neighbour lists, and lastly, (iv) apply a union-find algorithm to get a final set of clusters. Clustering was performed on approximately 1.5 billion unlabelled images, along with an evaluation on 3,385 labelled images. The main goal of the procedure was to group images into sets of near duplicates, but the total number of such sets in the 1.5 billion image dataset was unknown.

Tian et al. [41] proposed to tag faces with partial clustering and iterative labelling. In order to achieve high accuracy, most faces are remained un-grouped, and the sizes of face clusters are usually small. Therefore, many user operations are still needed to label all the faces.

Kapoor et al. [15] suggested integrating match/non-match pairwise priori constraints into active learning, in order to give the best face tagging order. To achieve satisfactory performance, this method relies on the amount of available priori constraints, whose number is usually limited under real face tagging scenarios.

2.1.1 Clustering Algorithms for Face Recognition

Zhao et al. [52] clustered personal photograph collections. Their approach combines a variety of contextual information including time based clustering, and the probability of faces of certain people to appear together in images, with identity estimates obtained via a 2D HMM and hierarchical clustering results based on body detection.

Cui et al. [8] developed a semi-automatic tool for annotating photographs, which employs clustering as an initial method for organizing photographs. First, LBP features are extracted from detected faces, and after, colour and texture features are extracted from detected bodies. Spectral clustering is performed, and the clustering results can then be manually adjusted by a human operator. Tian et al. [41] further developed this approach by incorporating a probabilistic clustering model, which incorporates a "junk" class, allowing the algorithm to discard clusters that do not have tightly distributed samples.

Zhu et al. [53] developed a dissimilarity measure based on the rankings of two faces being compared in each face's nearest neighbour lists, and transitively merged images

into clusters when the similarity, resulting rank-order distance function, is above some threshold. The feature representation used is the result of unsupervised learning [4]. In one hand, Wang et al. [45] primarily develop an approximate k-NN graph construction method; in one of their experiments they apply this method to construct the nearest neighbour lists required by [53], and use the rank-order distance measure to produce an improved k-NN graph (but do not perform hard assignment of faces into clusters). In the other hand, Otto et al. [25] further modified the algorithm presented in [53] by, (i) using deep representations of images (ii) considering only the absence and presence of an approximate nearest neighbours and (iii) transitively merging only once.

Shi et al. [34] proposed a face clustering method, called Conditional Pairwise Clustering (ConPaC) to group the face collection according to their hidden class (subject identity) using the pairwise similarity between face image. Instead of learning new similarity measures or representations and feeding them to a standard partitional or hierarchical clustering method, ConPac directly treats the adjacency between all pairs of faces as the variables to predict and look for a solution that maximizes the joint posterior probability of these variables given their corresponding pairwise similarity. To model this conditional distribution, they proposed a triplet consistency constraint which reveals such a dependency between the output variables that a valid adjacency matrix must be transitive to represent a partitional clustering. To model the problem, they used Conditional Random Field (CRF) and employ Loopy Belief Propagation to arrive at a valid adjacency matrix.

Yang et al. [48] proposed learning deep representations and image clusters jointly in a recurrent framework. Each image is treated as separate clusters at the beginning, and a deep network is trained using this initial grouping. Deep representation and cluster members are then iteratively refined until the number of clusters reached the predefined value.

Lin et al. [20] first used face images to pass them through a pre trained face DCNN model to extract the deep features. Then, they compute what they call 'Proximity-Aware similarity' scores using linear SVMs trained with corresponding neighbourhoods of the samples. Finally, the agglomerative hierarchical clustering method is applied on the similarity scores to determine the cluster labels to each sample.

2.2 Face Representation

Learning invariant and discriminative feature representation is the first step for a face verification and recognition system. It can be broadly divided into two categories: (i) hand-crafted features, and (i) feature representation learned from data.

In the first category, the first and most popular face recognition method is Eigenface [42] which was proposed in 1991. After, Ahonen et al. [1] showed that the Local Binary Pattern (LBP) is effective for face recognition. Gabor wavelets [47] [51] have also been widely used to encode multi-scale and multi-orientation information for face images. Chen et al. [5] demonstrated good results for face verification using the high-dimensional multi-scale LBP features extracted from patches around facial landmarks. Then, various local feature based methods emerged and they were naturally used by combining with the above linear models, such as Gabor+LDA [21], LBP+LDA [19] and so on and forth.

In the second category, Patel et al. [29] and Chen et al. [7] [35] applied dictionary-based approaches for image and video-based face recognition by learning representative features from the data which are compact and robust to pose and illumination variations. [27] [6] used the Fisher Vector encoding to generate over-complete and high-dimensional feature representation for still and video-based face recognition. Lu et al. [13] proposed a dictionary learning frame-work in which the sparse codes of local patches generated from local patch dictionaries are pooled to generate a high-dimensional feature vector.

The high-dimensionality of feature vectors makes these methods hard to train and scale to large datasets. However, advances in deep learning methods and the successful applications of CNN in image classification, it becomes the mainstream in the field of face recognition rapidly and have shown that compact and discriminative representation can be learned using DCNN from very large datasets.

Recently and due to the seminal 2012 paper by Krizhevsky et al. [17], that won the ImageNet competition [30], deep learning and convolutional neural networks in particular have been the focus of many researchers. Taigman et al. [39] learned a DCNN model on the frontalized faces generated with a general 3D shape model from a large-scale face dataset and achieved better performance than many traditional face verification methods. Sun et al. [37] [38] achieved results that surpass human performance for face verification on the LFW dataset using an ensemble of 25 simple DCNN with fewer layers trained on weakly aligned face images from a much smaller dataset than the former. Schroff et al. l[32] adapted the state-of-the art deep architecture for object recognition to face recognition and trained it on a large-scale unaligned private face dataset abandoning the traditional classification layer and instead introduced the triplet loss to directly learn an embedding space where feature vectors of different identities could be separated with Euclidean distance. This method also achieved top performances on face verification problems. Sankaranarayanan et al. [31], learnt a discriminative embedding using triplet probability constraints to address the unconstrained face verification problem for both verification and clustering problems.

These works essentially demonstrate the effectiveness of the DCNN model for feature learning for detection, recognition, verification and clustering of applications involving some sort of face feature extraction.

Chapter 3

Methodology

In this chapter we introduce and discuss the process of clustering a set of face images providing detailed information about each system's component. Our system consists of two major parts: (i) feature extraction from face images, (ii) followed by the application of a clustering algorithm.

As depicted in figure 3.1, first we perform face alignment (this step include face detection, bounding box calculation and facial landmarks extraction) on the input images to align faces. We tested different face alignment techniques. Next, those aligned face images are passed through a deep convolutional neural network (DCNN) model in order to extract deep features. In this step we used three different DCNN, where two of them, namely Openface [2] DCNN and VGG-face [28] DCNN, are pre-trained models downloaded from their respective website, and the third one is a fine-tuned version of the VGG-face DCNN we fine-tuned using the CASIA-webface [49] dataset. Finally, in the last step, we apply a clustering algorithm to determine the cluster labels of each face image on the aforementioned deep features. As with previous steps, we tested different clustering algorithms. In this case, Rank-Order [53] [25] clustering algorithm and Hierarchical Clustering algorithm (HAC) [11] [18] with different linkage criteria (Single Link, Complete Link, Ward and Average).

The following sections describe in detail each part of the system.

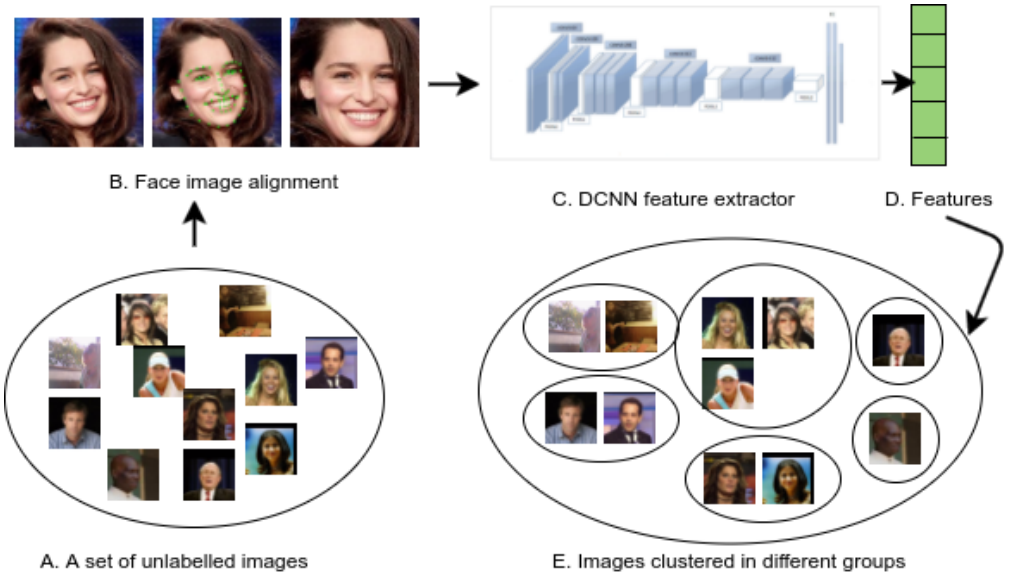


Figure 3.1. *Shown is an overview of the process for clustering a set of face images using a DCNN feature extractor. The first step (B) is to perform face alignment from the input images. Then, in the second step (C) a DCNN is used to extract deep features from every face image, leading to a vector representation of the face (D). Finally, the last step (E) is to apply a clustering algorithm to group similar faces together.*

3.1 Face Alignment

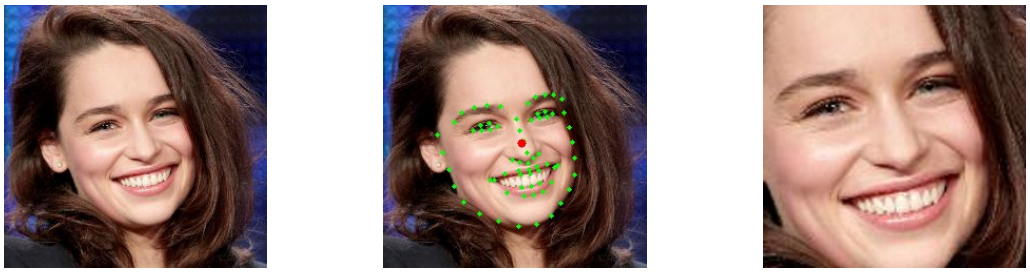
The feature extraction process is outlined in figure 3.1 (step B). Given an input, we use the DLIB [1] implementation of Kazemi and Sullivan's [16] ensemble of regression trees method to detect 68 facial landmarks - detecting facial landmarks is a subset of the shape prediction problem, where given an input image (and normally a ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape -. Then, image normalization is performed based on those detected key points, specifically we tested three different alignment techniques. Face alignment is done to let the DCNN extract a more adequate representation of the faces. Following is the explanation of all them:

Crop

This technique just crops the face around a point. Basically, from the detected facial landmarks (green points in figure 3.2) and the bounding box it calculates the central point

[1] http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html

(red point in figure 3.2) and then crops the image around this point with the specified width and height. In this case, there is no face transformation.

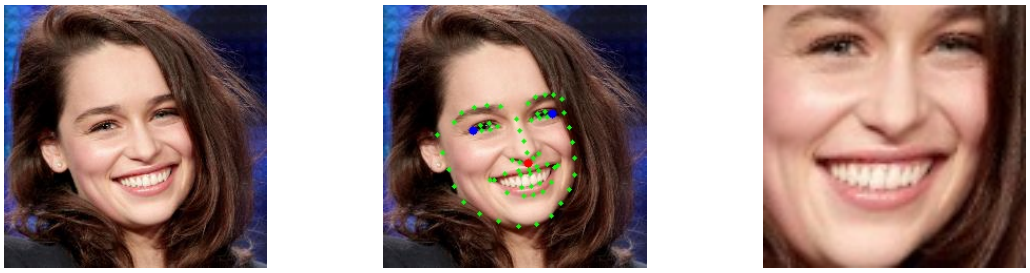


(a) **(b)** **(c)**

Figure 3.2. *A crop face image alignment example. (a) shows the original image; (b) shows the 68 landmark point detected by [16], where the red point is used as a central point to crop the image around it; and (c) is the final aligned face image.*

Point

In this case, we are using the alignment that accompany Openface library [2]. It works as follows: (i) after localizing the facial landmarks (green points in figure 3.3), as decried in the previous section, use OpenCV's affine transformation to try to make the outer eyes (blue points in figure 3.3) and nose (red point in figure 3.3) appear in the same location on each image. The red point is also used to crop the face image with the specified width and height.

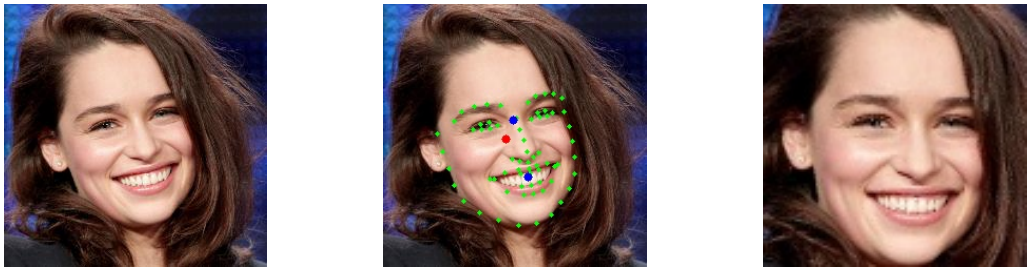


(a) **(b)** **(c)**

Figure 3.3. *A crop face image alignment example. (a) shows the original image; (b) shows the 68 landmark point detected by [16], where the red point is used to centre the cropped face image and the blue points are used to centre the face along the x-axis; and (c) is the final aligned face image.*

Mass

This technique is the one that [44] uses. It works as follows: (i) after localizing the facial landmarks (green points in figure 3.4), as described in the previous section; (ii) rotate the face in the image plane to make it upright based on two centre eye positions (right and left eye). The centre points of the eyes are found by averaging all the landmarks in the eyes regions; iii) find a central point on the face by taking the mid point between the leftmost and rightmost landmarks (red point in figure 3.4); iv) centre the faces in the x-axis, based on the central point; v) find the eye and mouth centre point (blue points in figure 3.4) by averaging over all the landmarks in the eye and mouth regions, respectively; vi) the eye centre point is placed at 45% of image height from the top of the image while the mouth centre point is placed at 25% of image height from the bottom of the image. One problem we should note, is that the midpoint (red point in figure 3.4) is not consistent across pose. If some faces exhibit significant yaw, the computed midpoint will be slightly different from the one computed in frontal faces.



(a) **(b)** **(c)**

Figure 3.4. *A crop face image alignment example. (a) shows the original image; (b) shows the 68 landmark point detected by [16], where the red point is used to centre the face image along the x-axis and the blue points are used to centre the face along the y-axis as well as to crop the image; and (c) is the final aligned face image.*

3.2 Face Representation

Since our goal is to cluster people's faces captured under unconstrained conditions, i.e. variations in illumination, facial expressions, occlusion, age, pose, etc. we opted to use a DCNN for face embeddings (representation) following the success of such methods on the LFW [12] benchmark [2]. This benchmark shows that a lot of deep learning approaches have been successfully applied to face recognition/classification and some of them to clustering

[2] http://vis-www.cs.umass.edu/lfw/results.html

face images. However, most of them leverage private training sets. In this work, we are using two different pre-trained DCNNs, namely the one from the Openface library [2] and the other one is the VGG-face [28] DCNN. Further, we fine-tune the VGG-face network with the CASIA-webface [49] dataset. Next sections explain them.

Openface

Figure 3.5 shows the layer description of the Openface [2] neural network model, which is a modified version of the NN4 model from Facenet [32] hand-tuned to have less parameters for the smaller training dataset. Each row is a layer in the neural network and the last six columns indicate the parameters of pooling or the inception layers. Dimensionality reductions to N dimensions after pooling is denoted with "Np". The normalization used is the local response normalization.

Unlike original Facenet [32] which was trained using a private dataset with 100M-200M images, Openface was trained with only 500k images combining two large labelled face recognition datasets for research, CASIA-WebFace [49] and FaceScrub [24].

The input to Openface's network is a 96x96 RGB image and the output is a 128-dimensional vector representing the face image.

type	output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
conv1 ($7 \times 7 \times 3, 2$)	$48 \times 48 \times 64$						
max pool + norm	$24 \times 24 \times 64$						m 3×3, 2
inception (2)	$24 \times 24 \times 192$		64	192			
norm + max pool	$12 \times 12 \times 192$						m 3×3, 2
inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	m, 32p
inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	ℓ_2, 64p
inception (3c)	$6 \times 6 \times 640$		128	256,2	32	64,2	m 3×3, 2
inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	ℓ_2, 128p
inception (4e)	$3 \times 3 \times 1024$		160	256,2	64	128,2	m 3×3, 2
inception (5a)	$3 \times 3 \times 736$	256	96	384			ℓ_2, 96p
inception (5b)	$3 \times 3 \times 736$	256	96	384			m, 96p
avg pool	736						
linear	128						
ℓ_2 normalization	128						

Figure 3.5. *Details of the face CNN configuration model. For each convolution/inception layer, the filter size, number of filters, stride and padding are indicated. Image taken from [2].*

VGG-face

Figure 3.6 shows the layer description of the VGG-face [28] neural network model. It is based on the VGG-Very-Deep-16 CNN architecture [36]. In [28], the authors describe different configuration. Configuration A, the one depicted in figure 3.6), comprises 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and max-pooling. The first eight such blocks are said to be convolutional as the linear operator is a bank of linear filters (linear convolution). The last three blocks are instead called Fully Connected (FC); they are the same as a convolutional layer, but the size of the filters matches the size of the input data, such that each filter "senses" data from the entire image. All the convolution layers are followed by a rectification layer (ReLU), but they do not include the Local Response Normalisation operator. In this work, we are using the configuration B which is similar to A but include 2 additional convolution layers.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Figure 3.6. *Details of the face CNN configuration model A [28]. For each convolution/inception layer, the filter size, number of filters, stride and padding are indicated. The FC layers (last three) are listed as "convolution". Image taken from [28].*

The CNN configuration A was trained from scratch in a large public dataset from the same authors (VGG-face dataset [28]) with 2.6M images. Whereas configurations B was trained by starting from the trained A using the same dataset. This is obtained by appending additional fully connected layers to A, randomly initialised, and then training the latter (with a lower learning rate) again.

The input to the network is an RGB face image of size 224×224 with the average face image (computed from the training set) subtracted – this is critical for the stability of the optimisation algorithm. The output is a 4096-dimensional vector representing the face image.

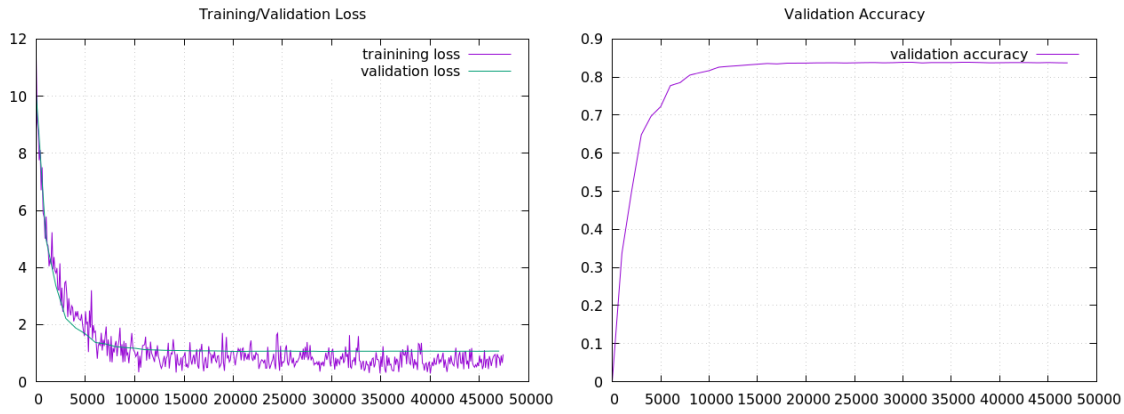
VGG-face fine-tuned

In practice, researchers either learn their deep models from scratch or fine-tune only the last few layers an existing pre-trained model. Usually, the fine-tune process is done in existing networks that are trained on a large dataset like the ImageNet (1.2M labelled images) by continue training it on a smaller dataset of our own. Provided that our dataset is not drastically different in context to the original dataset, the pre-trained model will already have learned features that are relevant to our own problem.

Thus, in order to check if the performance of our system improves, we fine-tune the VGG-face [28] DCNN model using the CASIA-webface [49] dataset. We split this dataset such that, 80% of the images are for training and 20% are for validation (see section 4.1 for actual numbers).

The network is optimised using stochastic gradient descent using min-batches of 64 samples and momentum coefficient of 0.9. The model is regularised using dropout and weight decay; the coefficient of the latter was set to 0.0005 whereas the former was applied to fully connected layers with a rate of 0.5 due to the large number of parameters. The learning rate was initially set to 0.001 and then decreased by factor of 5 every 2500 iterations. The network was trained for a total of 50k iterations. The learning rate was set for only the last fully connected layer (FC8) while the remaining layers of the network didn't change.

Figure 3.7 shows training and validation loss and validation accuracy. As we can see, both training and validation loss starts high and decreases over iterations while the network is learning. In the other hand, the validation accuracy starts being small and every time the learning rate changes, the accuracy increases. This is the scenario during a long number of iterations. When the learning rate reaches very small values, i.e. $1e^{-7}$, the network no longer learns and remains stable during the remaining iterations. This can be solved by, (i) performing some data augmentation to increase the size of the data, so the network has more data to learn from, (ii) reducing the number of iterations the learning rate changes, and (ii) instead of freezing the whole network but the last FC layer, try to freeze only the first convolutional layers and let the network learn the remaining layers. Due to a lack of resources and time, we were unable to test all this different configurations and we will let it for a possible future line of research.



(a) **(b)**

Figure 3.7. *Model fine-tuning: loss and accuracy for training and validation phases.*

3.3 Clustering

A large number of clustering methods have been proposed in the literature based on squared-error, mixture models, nearest neighbour and graph-theoretic approaches [14]. In this work, we study different clustering algorithms, namely: approximate rank-order clustering algorithm [53] [25], hierarchical agglomerative clustering algorithm (HAC) [11] [18] with different linkage methods; how to compute the distance *d(s, t)* between two clusters *s* and *t*. In our experiments, we used four methods: Single link (SL), Complete link (CL), Ward (WD) and Average (AG). The last clustering algorithm we will use is the so-famous *k*-means for the egocentric dataset as a baseline, since (i) it is perhaps the most well-known clustering algorithm, (ii) has only a few parameters to tune.

These algorithms are described in detail in next sections.

3.3.1 Approximate Rank-Order Algorithm

In this section, we present the original algorithm by [53] in detail and then the modified version [25] of it.

Rank-Order Algorithm

The rank-order clustering algorithm proposed by Zhu et al. [53], is a form of agglomerative hierarchical clustering, using a nearest neighbour based distance measure. The

procedure of an agglomerative hierarchical clustering is as follows: given some distance metric, initializes all samples to be separate clusters and then iteratively merge the two closest clusters together. This requires defining a cluster-to-cluster distance metric. In the algorithm, the distance between two clusters is the minimum distance between any two samples in the clusters.

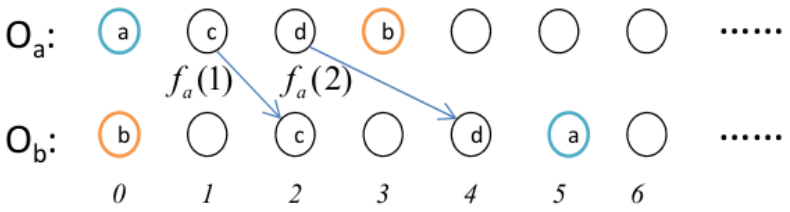


Figure 3.8. *Rank-Order distance example: O_a and O_b are two order lists which are ranked using face a and b. The asymmetric distance $D(a,b) = O_b(f_a(0)) + O_b(f_a(1)) + O_b(f_a(2)) + O_b(f_a(3)) = O_b(a) + O_b(c) + O_b(d) + O_b(b) = 5+2+4+0 = 11$. Image taken from [53]*

Given two faces a and b, first the algorithm generates two order lists O_a and O_b by sorting faces in the dataset according to absolute distance, as shown in figure 3.8. Next, the first distance metric defines an asymmetric Rank-Order distance $D(a,b)$ between a and b as follows:

$$D(a,b) = \sum_{i=0}^{O_a(b)} O_b(f_a(i)) \tag{3.1}$$

Where $f_a(i)$ gives the i^{th} face in the order (neighbour) list of a. For example, $f_a(1)$ refers to face c, the nearest one to face a in in figure 3.8. $O_b(f_a(i))$ returns the ranking order of the face $f_a(i)$ in b's order list. $O_a(b)$ is the order of face b in a's order list. Basically, this distance is the summation of rank orders of a's top neighbours in b's order list as depicted in figure 3.8.

This asymmetric distance function is further normalized to define a symmetric distance between two faces, a and b, as follows:

$$D^R(a,b) = \frac{D(a,b) + D(b,a)}{min(O_a(b), O_b(a))} \tag{3.2}$$

where $min(O_a(b), O_b(a))$ is a normalization factor to make the distance comparable. This normalization is important since $D(a,b)$ is biased towards penalizing large $O_a(b)$. $D(b,a)$ is defined by switching the roles of a and b. A small distance means many a's top neighbours are also b's top neighbours and vice-versa.

After distances are computed, clustering is performed as follows: (i) initialize every face to its own cluster, (ii) then computing the symmetric distances between each cluster merging any clusters with distance below a threshold. After, (iii) nearest neighbour lists for any newly merged clusters are merged and distances between the remaining clusters are computed again iteratively, until no further clusters can be merged.

Approximate Rank-Order Algorithm

As we saw, the rank-order clustering algorithm has a scalability problem in that it requires computing nearest neighbour lists for every instance in the dataset, which has an $O(n^2)$ cost. Although various approximation methods exist for computing nearest neighbours, they are typically only able to compute a short list of the top k nearest neighbours efficiently, rather than exhaustively ranking the dataset.

In order to apply approximation methods for faster nearest neighbour computation, it was a requirement to apply some modifications of the original clustering algorithm. In particular, rather than considering all the neighbours in equation 3.1, it is only necessary to sum up to at most the top k neighbours. Moreover, if only a short list of the top k- neighbours is considered, the presence or absence of a particular neighbour on the short list may be more significant than the neighbour's numerical rank (the order). Thus, the asymmetric distance of this algorithm is based on directly summing the presence/absence of shared nearest neighbours rather than the ranks as the original algorithm does. Then, equation 3.1 is re-defined as follows:

$$D_m(a,b) = \sum_{i=0}^{min(O_a(b),k)} I_b(O_b(f_a(i)),k) \tag{3.3}$$

where $I_b(x,k)$ is an indicator function with a value of 0 if face x is in face b's top k nearest neighbours, and 1 otherwise.

The normalized symmetric distance used in the original algorithm (3.2) is still effective and contributes to more accurate clustering results. Further, the normalized distance measure with the modifications done is defined as follows:

$$D_m^R(a,b) = \frac{D_m(a,b) + D_m(b,a)}{min(O_a(b), O_b(a))} \tag{3.4}$$

In addition to the modification carried on to the asymmetric distance measure, following are some modifications done to improve the runtime of the algorithm: (i) we only compute distances between samples which share a nearest neighbour, and (ii) we only perform one round of merges of individual faces into clusters, as described by the authors.

The clustering algorithm is done following the next steps: (i) compute a set of the top k-nearest-neighbours for each face in the dataset. After the list is computed, (ii) compute pairwise distances between each face and its top k-nearest-neighbour lists following equation 3.4. Finally, after all distances are computed, (iii) transitively merge all pairs of faces with distances below a threshold.

This clustering algorithm takes as input two parameters that need to be tuned. (i) A cut-off parameter, called t, which acts as a distance threshold (below which any two faces should form a cluster or join an existing one) and (ii) a parameter, called k, which picks the top k-neighbour for every face to build the nearest-neighbour list. Thus in our experiments, we vary these two parameters over small range and evaluate the resulting clustering (using the metrics explained in section 4.2. We pick the result that yields the best result.

3.3.2 Agglomerative Hierarchical Clustering

Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (known as dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample. Generally, there are two types of hierarchical clustering:

- Agglomerative: This is a "bottom up" approach: each sample starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: This is a "top down" approach: all samples start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In this work, we are using the "bottom up" approach: Agglomerative hierarchical clustering (HAC) [11] [18]. As described earlier, each face starts forming its own cluster, and clusters are successively merged together depending on the criteria used (linkage). The linkage criteria determines the metric used for merge strategy. Here, we use the following linkage criteria to compute the distance $d(s,t)$ between two clusters s and t.

- Single linkage: assigns

$$d(u,v) = min(dist(u[i],v[j]))$$

for all points i in cluster u and j in cluster v. This is also known as nearest point algorithm.

- Complete linkage: assigns

$$d(u,v) = max(dist(u[i],v[j]))$$

for all points i in cluster u and j in cluster v. This is also known as farthest point algorithm.

- Average linkage: assigns

$$d/u,v) = \sum_{ij} \frac{d(u[i],v[i])}{(|u| * |v|)}$$

for all points i and j where $|u|$ and $|v|$ are the cardinalities of clusters u and v, respectively. This is also known as UPGMA algorithm (Unweighted Pair Group Method with Arithmetic Mean).

- Ward linkage: uses the Ward variance minimization algorithm. $d(u,v)$ is computed as follows:

$$d(u,v) = \sqrt{\frac{|u|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 + \frac{|v|}{T}d(s,t)^2}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $|*|$ is the cardinality of its argument. This is also known as the incremental algorithm.

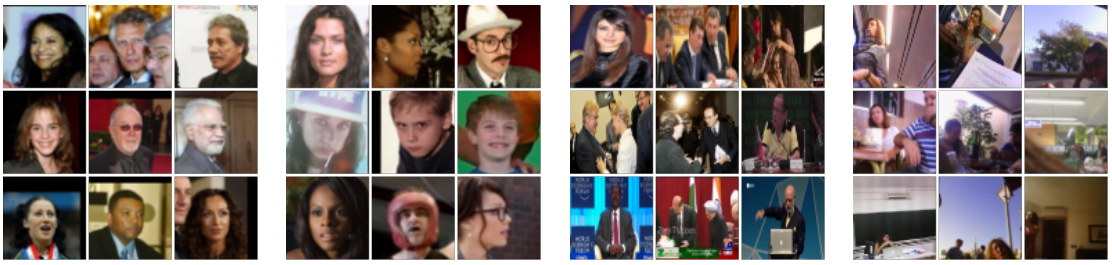
This clustering algorithm takes as input one parameter that need to be tuned. A cut-off parameter, called h, which acts as a distance threshold (below which any two faces should form a cluster or join an existing one). Thus in our experiments, we vary this parameter over small range and evaluate the resulting clustering (using the metrics explained in Section 4.2. We pick the result that yields the best result.

Chapter 4

Evaluation and Experiments

4.1 Datasets

We used several unconstrained face datasets in our experiments, namely, CASIA-webface dataset [49] for fine-tuning the VGG-face deep network feature representation, the Labelled Faces in the Wild (LFW) dataset [12] and IARPA Janus Benchmark-B (IJB-B) dataset [46] for cluster evaluation. Finally, an egocentric dataset, Egocentric Dataset of the University of Barcelona – Objects (EDUB-Obj) dataset [3] were used as an application to show how to cluster faces in egocentric images. An example of face images from each dataset are shown in figure 4.1.



(a) **(b)** **(c)** **(d)**

Figure 4.1. *Example face images from the a) LFW, b) CASIA-webface, c) IJB-B, and d) EDUB-Obj datasets.*

CASIA-webface

The CASIA-webface dataset [49] is a semi-automatically collected face dataset to push the development of the face recognition domain. It contains $494,414$ images of $10,575$

subjects (mostly of them being celebrities) downloaded from internet. However, we were unable to localize faces in some of the images with the face detector in Dlib library [1]. Thus, we use a subset (the detected faces) of 456,848 face images of the 10,575 subjects to fine-tune the VGG-face network. Further, this dataset was split into training and validation sets with 80% (365,478 face images) of the data for training and 20% (91,370 face images) for validation. This dataset has been popular for training deep networks.

LFW

LFW [12] is a collection of 13,233 face images of 5,749 individuals downloaded from the web. Face images in this dataset contain significant variations in pose, illumination, and expression. However, the images in this dataset were selected only if a face could be detected by the Viola-Jones detector [43]. The dataset was constructed by searching for images of celebrities and public figures. One issue is that the distribution of images per subject is quite imbalanced in this dataset. Indeed, of those 5,749 individuals, 4,069 have only one face image each. Thus, only 1,680 individuals (classes) containing more than one face. Since we cannot assume real-world datasets will be well balanced, although we could construct a subset of LFW with more balanced clusters, we do experiments on the entire LFW dataset.

IJB-B

The IJB-B dataset [46] is composed of different protocols to evaluate different scenarios, i.e. classification, verification and clustering. In this case, we are interesting in clustering protocols. Under clustering protocol, two protocols are defined for IJB-B dataset: (i) clustering of detected faces and (ii) face detection + clustering. Since we are focusing in the task of face clustering, we will use the first protocol and assume faces have already been detected. Thus, in the first protocol, we find seven different experiments with increasing number of subjects. These experiments, involve respectively, 32, 64, 128, 256, 512, 1,024 and 1,870 subjects with total of 1,473, 2,566, 4,793, 11,186, 19,583, 37,653 and 68,714 images, respectively. The author's designed these tests to test an algorithm's ability to identify multiple instances of the same subject from a collection of various pieces of images and video frames. For each test, all imagery for each selected subject in IJB-B is used (still images and video frames) and is a superset of the previous test (e.g., IJB-B-64 contains all imagery from IJB-B-32 plus imagery from 32 additional subjects and so on).

[1] http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html

Many images in the IJB-B datasets are in extreme poses or of low quality (images captured in various conditions, such as: video/photo, indoor/outdoor, pose, expression and illumination), making the clustering task more difficult for IJB-B than for LFW.

EDUB-Obj

EDUB-Obj [3] is a dataset for object localization or segmentation composed of a total of $4,912$ images acquired by the wearable camera Narrative, which captures images in a passive way every $30-60$ seconds. The dataset is divided in 8 different days which capture daily life activities like shopping, eating, riding a bike, working, attending meetings, commuting to work, etc. It has been acquired by 4 different subjects, and each of them having captured 2 different days. We can find a total of $11,281$ different objects grouped in 21 different classes in the whole dataset. Of those classes, the one we are interested in is the face class; which has only 565 faces in total. The egocentric images are in extreme conditions, such as different poses, face occlusions (partial/complete), low quality of illumination, different expressions and they are taken in people's daily life activities, making face clustering in this type of dataset way harder than the IJB-B or LFW datasets.

4.2 Clustering Evaluation

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define a well separations of the data. If we talk about supervised clustering, it is possible to check if a cluster is similar to some ground truth set of classes (they are called external metrics). In the other hand, if we talk about unsupervised clustering, we can see if a cluster satisfies some assumption such that members belong to the same class are more similar that members of different classes according to some similarity metric (they are called internal metrics).

In this work, we are using both types of clusters, supervised one (we know the labels of each face in the dataset) to evaluate the performance of the different models, and unsupervised one (there are no labels to identify each face in the dataset) to cluster egocentric images. Thus, we will use F_1-score, defined by pairwise precision/recall, as the external metric, for the labelled datasets and Silhouette score, as internal metric, for the egocentric (unlabelled) dataset.

F_1-score measure

In order to define this validation metric, we must define two other important metrics: precision and recall. All these metrics are summarized below:

Pairwise precision (P_{pair}) is defined as the fraction of the number of pairs within the same cluster which are of the same class, over the total number of same-cluster pairs. While, pairwise recall (R_{pair}) is defined as the fraction of the number of pairs within a class which are placed in the same cluster, over the total number of same-class pairs. These measures capture two types of error, a clustering which places all samples as individual clusters will have high precision, but low recall, while a clustering which places all samples in the same cluster will have high recall, but low precision. Thus, these two measures can be summarized in the F-measure. Using these metrics, the F_1-score is computed as:

$$F_1 = \frac{2 * P_{pair} * R_{pair}}{P_{pair} + R_{pair}} \tag{4.1}$$

Silhouette score

When the ground truth labels are not known (or not provided), evaluation must be performed using the model itself. The Silhouette Coefficient is an example of such an evaluation, where a higher Silhouette Coefficient score relates to a model with better defined clusters. The Silhouette Coefficient is defined for each sample and is composed of two scores:

- a: The mean distance between a sample and all other points in the same class.
- b: The mean distance between a sample and all other points in the next nearest cluster.

Then the Silhouette Coefficient s_i for a single sample is then given as:

$$s_i = \frac{b - a}{max(a, b)} \tag{4.2}$$

Finally, in order to compute the Silhouette score, S, for all samples, we must average over all Silhouette Coefficient for each sample, as follows:

$$S = \frac{1}{N} \sum_{i=0}^{N} s_i \tag{4.3}$$

Where N is the total number of samples. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. This metric has the advantage that the score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

4.3 Experimental Results

This section describes all the experiments carried on during the development of this work with different features extractors (Openface, VGG-face and VGG-face-finetuned), different clustering algorithms (rank-order, HAC-SL, HAC-CL, HAC-WD, HAC-AG, and k-means) and different face alignment methods (point, mass and crop). Table 4.1 shows which parameters were used to test different algorithms.

Algorithm	Parameter	Start	End	Step
Rank-Order	k	5	150	5
	t	0.05	3	0.05
HAC	h	0.1	5	0.01
kMeans	k	2	20	2

Table 4.1. *Values for different parameters and algorithms tested during the development of this work. The step is the increment value from start to end (both inclusive). For instance, for rank-order: parameter k was tested with 5, 10, 15, and so on until 150.*

4.3.1 LFW Dataset Experiments

Using: Rank-Order Clustering Algorithm

Table 4.2 shows the results when running the rank-order clustering algorithm on the entire LFW dataset. Each sub-table uses three different techniques for face alignment and two different features extractor. The first column shows the results using features extracted with the Openface DCNN, while the second column shows the results using features extracted with the VGG-face DCNN. Lastly, the third column shows the results using our VGG-face fine-tuned model.

Alignment	Openface	VGG-face	VGG-face-finetuned
Point	**0.290**	0.444	0.555
Mass	0.009	0.684	**0.738**
Crop	0.062	**0.729**	0.679

Table 4.2. *Comparison of the F_1 score of the Rank-Order clustering algorithm on LFW dataset. In bold the best result within each DCNN. In blue the best overall result.*

As we can see, in table 4.2, the best result among the three face alignment techniques using Openface features is given by the *Point* technique because this DCNN was trained using this face alignment technique, and thus, if we align faces in different way, the extracted features will be slightly different. In the other hand, using VGG-face features, the best result is given by the *Crop* technique. This is, because this DCNN was trained without any kind of face alignment, the face images were only cropped. Thus, the reason of why the other techniques fail to give better result, is because of the clustering algorithm itself. In the last column of table 4.2 is given the result of our fine-tuned VGG-face network. As we can see, it gives the overall result in this clustering algorithm using *Mass* face alignment technique.

To Summarize, the combination of VGG-face-finetuned features and *Mass* technique gives the better result (**0.738**), in comparison to the other feature extractor and other face alignments, for this clustering algorithm. We will use this result for further comparisons in next sections.

Using: HAC Clustering Algorithm

Table 4.3 shows the results when running the HAC clustering algorithm on the entire LFW dataset. As in previous comparison, we use three different techniques for face alignment and two different feature extractor. To use this clustering algorithm, we must specify a linkage strategy (see 3.3). In our experiments, we use the following methods: Single link (SL), Complete link (CL), Ward (WD) and Average (AG) (see 3.3 for more detail). Table 4.3a shows the result using features extracted with the Openface DCNN, while table 4.3b shows the result using features extracted with the the VGG-face DCNN.

Alignment	SL	CL	WD	AG
Point	0.520	0.461	0.346	**0.666**
Mass	0.006	0.012	0.013	0.011
Crop	0.099	0.035	0.039	0.053

(a) *Using Openface DCNN model.*

Alignment	SL	CL	WD	AG
Point	0.664	0.448	0.202	0.636
Mass	0.903	0.804	0.334	**0.938**
Crop	0.862	0.802	0.318	0.905

(b) *Using VGG-face DCNN model.*

Alignment	SL	CL	WD	AG
Point	0.6924	0.4108	0.2452	0.6935
Mass	0.8626	0.8278	0.3472	0.9179
Crop	0.9035	0.7026	0.3768	**0.9227**

(c) *Using VGG-face-finetuned DCNN model.*

Table 4.3. *Comparison of the F_1 score of the HAC clustering algorithm with different linkage criteria on LFW dataset. In bold the best result within each DCNN. In blue the best overall result.*

As in previous section, the *Point* face alignment technique using Openface DCNN features extractor gives best result among the other techniques (as we can see in table 4.3a). Besides, the *Average* linkage criteria yields best result among other criteria. This is due to the fact that *Average* criteria calculates its distance between two clusters using the mean distance between all points in both clusters. In the other hand, as we see in table 4.3b, the *Mass* technique gives best result among other face alignment techniques. As stated in previous section, this is because the network was trained without any type of alignment. Indeed, *Mass* technique gives better face alignment than the other two, because it takes into account several facial landmarks (see section 3.1). *Average* linkage criteria still being the best in comparison to the other.

If we have a look at table 4.3c, our fine-tuned model outperforms Openface model in all cases and it is able to outperform some VGG-face results. But the best overall result is done by VGG-face with a small difference from our fine-tuned model.

Thus, the combination of VGG-face features, *Mass* face alignment technique and *Average* linkage criteria, yields to better result (**0.938**). We will use this result for further comparisons in next sections.

RankOrder vs HAC

Comparing table 4.2 and table4.3, we can see that using VGG-face as a face features extractor and HAC-Average as a clustering algorithm, clearly outperforms Openface as a face feature extractor and RankOrder as a clustering algorithm. Thus, to compare with state of the art results using this dataset, i.e. LFW, we will be using the aforementioned combination.

4.3.2 IJB-B Dataset Experiments

The result of the six (in total there are seven, but we were unable to execute the last one due to a lack of resources) experiments in this dataset are shown in table 4.4. As in previous sections, we used both DCNN to extract face features in all the the experiments. In this case we are not using any sort of face alignment (we tested the same face alignments as in previous sections, but they yield to a very bad result). Thus we are using the *Crop* technique, which only crop the face specified by a bounding box (the bounding box is taken from the provided one along the dataset).

In all the experiments, the best result is given by the VGG-face DCNN and HAC-*Average* clustering algorithm. And as expected, with an exception of the firsts experiments, as the number of identities increases, the F_1 score decreases.

Algor.	OF	VGG	VGG-tune
RO	0.239	0.679	0.728
SL	0.307	0.916	0.698
CL	0.199	0.907	0.747
WD	0.238	0.0667	0.064
AG	0.284	**0.919**	0.853

(a) *IJB-B-32*

Algor.	OF	VGG	VGG-tune
RO	0.209	0.760	0.684
SL	0.212	0.804	0.555
CL	0.162	0.829	0.703
WD	0.184	0.622	0.588
AG	0.207	**0.887**	0.721

(b) *IJB-B-64*

Algor.	OF	VGG	VGG-tune
RO	0.498	0.596	0.530
SL	0.545	0.902	0.635
CL	0.232	0.899	0.696
WD	0.201	0.438	0.457
AG	0.408	**0.939**	0.868

(c) *IJB-B-128*

Algor.	OF	VGG	VGG-tune
RO	0.316	0.614	0.588
SL	0.308	0.891	0.831
CL	0.129	0.854	0.696
WD	0.154	0.401	0.457
AG	0.259	**0.932**	0.868

(d) *IJB-B-256*

Algor.	OF	VGG	VGG-tune
RO	0.255	0.574	0.584
SL	0.187	0.791	0.710
CL	0.103	0.741	0.693
WD	0.102	0.420	0.401
AG	0.175	**0.847**	0.751

(e) *IJB-B-512*

Algor.	OF	VGG	VGG-tune
RO	0.196	0.516	0.570
SL	0.096	0.688	0.533
CL	0.069	0.659	0.441
WD	0.075	0.385	0.369
AG	0.122	**0.794**	0.700

(f) *IJB-B-1024*

Table 4.4. *Comparison of the F_1 score of Rank-Order and HAC (with different linkage criteria) clustering algorithm on IJB-B dataset. In bold the best result using each DCNN.*

4.4 Comparison to State of the Art

In this section we will compare recent state of the art works with our results. We will compare with the following works:

Otto et at. [26] they use a COTS (commercial off the shelf) matcher to extract features from the faces. They didn't give the name of the method due to licensing agreements, but

is one of the top performing algorithms in the NIST FRVT 2014 evaluations (listed as COTS). For clustering algorithm, they use Rank-Order algorithm.

In [25] they use a CNN with 10 convolutional layers trained from scratch on CASIA-webface dataset. For clustering, they use approximate Rank-Order algorithm.

While in [34] they use a Res-net with 50 layers trained on a combination of VGG-Face dataset and CASIA-webface. For clustering, they create their own approach, which it is called ConPac (Conditional Pairwise Clustering). This approach models the problem of clustering as a CRF (Conditional Random Field) with Loopy Belief Propagation.

In [31] they use a CNN with 7 convolutional layers trained from scratch on CASIA-webface dataset. After training, they remove the last layer (soft-max) and introduce a new layer (Triplet Probabilistic Embedding) to learn a discriminative embedding for the faces. For clustering, they use HAC (Hierarchical agglomerative clustering) with average linkage criteria.

In our case, our best result is obtained using the VGG-Face DCNN descriptor (Softmax version). This DCNN is based on the VGG-Very-Deep-16 and it is trained in the VGG-Face dataset. We are using a pre-trained model downloaded from VGG website [2]. Although we fine-tuned this network with a mid-large dataset, we are unable to outperform VGG-face. The reason may be due to the training parameters, that need to be tuned a little bit more. Also regarding the dataset used for fine-tuning, we need to use some sort of data augmentation to let the network learn more and generalise well. As for clustering, our best result is obtained using the HAC (Hierarchical agglomerative clustering) with average linkage criteria.

Table 4.5 shows the F_1-score for the different works using the entire LFW dataset. The best result is obtained by [34], but as we can see we are relatively close to the state of the art results.

[2] http://www.robots.ox.ac.uk/~vgg/software/vgg_face

Method	F_1 **score**
[26]	0.390
[25]	0.870
[34]	**0.965**
[31]	0.955
Ours	0.938

Table 4.5. *F_1-score of different works performed on the LFW dataset.*

Table 4.6 compares the result of different clustering protocols defined by **IJB-B** dataset between out work and a state of the art results. As we can see, in the first two test, the best are from [34], nevertheless we are very close. Indeed, in the remaining tests we outperform the other methods. As we can see, the result of [34] decreases drastically a we increase of number identities. In comparison, our approach also decreases, as expected, but the amount of decrease is not that heavier than [34]. In the other hand, if we compare to the baseline results [12], which they were obtained from a government off the shelf (GOTS) algorithm, either the work of [34] and ours clearly outperform them.

Method	**IJB-B-32**	**IJB-B-64**	**IJB-B-128**	**IJB-B-256**	**IJB-B-512**	**IJB-B-1024**
Baseline [12]	0.395	0.396	0.445	0.446	0.401	0.403
[34]	**0.937**	**0.897**	0.814	0.459	0.424	0.348
Ours	0.919	0.887	**0.939**	**0.932**	**0.847**	**0.794**

Table 4.6. *F_1 score of different works performed on the IJB-B dataset.*

The reason of why we are comparing only to two works, is because this is a newly dataset and there is no much literature out there using this dataset for clustering. There were another dataset called IJB-A (actually it was a small dataset) and some works used this dataset to cluster faces, but it is no longer exist. It was replaced by this one and therefore we cannot compare to the ones using the previous version. The other reason is because a vast of literature are using this dataset (IJB-B) for tasks like face verification and face classification and thus we cannot compare with them, because we are performing clustering of faces.

4.5 Application: Clustering Egocentric Faces

In this practical example of clustering face images, we want to use an egocentric dataset, as stated in section 4.1. The data set we are using is the EDUB-Obj (see section 4.1 for details). Basically, this dataset is divided in different subjects taken egocentric images in different days. For instance 'Subject1_1' means person 1 in day 1. Thus, we have four distinct subjects taken egocentric pictures in two different days during their daily activity.

In our experiments, we used the same algorithms, same face alignment techniques and same feature extractors as explained in previous experiments. For every algorithm we tested different values of their respective parameters. To evaluate the performance of all this models and pick the best ones, we perform the Silhouette score (see section 4.2 for more details). In short, this score is bounded between -1, for very bad clustering results, and $+1$, for good clustering. Scores around 0 means that the clusters are overlapped.

Table 4.7 shows the results of different clustering algorithms using Openface DCNN features for every subject in the two different days. As we can see, most of the best results, in terms of silhouette score, are attained to HAC clustering algorithm. In fact, rank-order only outperforms HAC in one case by a small difference. In the other hand, k-means gives lower results than the other two algorithms.

Subject	Rank-Order	HAC	k-Means
Subject1_1	0.071 (43)	**0.202 (2)**	0.160 (2)
Subject1_2	0.080 (100)	**0.269 (2)**	0.160 (2)
Subject2_1	**0.228 (2)**	**0.228 (2)**	0.188 (2)
Subject2_2	0.187 (8)	**0.309 (2)**	0.241 (2)
Subject3_1	0.111 (67)	**0.283 (2)**	0.205 (5)
Subject3_2	**0.289 (4)**	0.249 (3)	0.249 (3)
Subject4_1	0.121 (56)	**0.225 (2)**	0.163 (2)
Subject4_2	0.113 (35)	**0.460 (2)**	0.253 (2)

Table 4.7. *Silhouette score for for different clustering algorithms using Openface DCNN features. In parenthesis is shown the number of clusters. In bold, best result for every subject.*

Table 4.8 shows the results of different clustering algorithms using VGG-face DCNN features for every subject in the two different days. As we can see, most of the best results, in terms of silhouette score, are attained, as in previous table, to HAC clustering algorithm.

In this case, rank-order don not outperform HAC, but in two cases they show similar result. k-Means, with this features, outperforms the other two algorithms in just one case by a small amount and equals them in other case.

Subject	Rank-Order	HAC	k-Means
Subject1_1	0.113(51)	**0.176 (18)**	0.157 (5)
Subject1_2	0.088 (48)	0.155 (9)	**0.165 (3)**
Subject2_1	**0.149 (2)**	**0.149 (2)**	0.112 (3)
Subject2_2	0.131 (11)	**0.179 (2)**	0.135 (6)
Subject3_1	0.142 (54)	**0.233 (15)**	0.216 (2)
Subject3_2	**0.225 (5)**	**0.225 (5)**	**0.225 (5)**
Subject4_1	0.140 (23)	**0.167 (16)**	0.149 (8)
Subject4_2	0.089 (2)	**0.253 (8)**	0.244 (4)

Table 4.8. *Silhouette score for for different clustering algorithms using VGG-face DCNN features. In parenthesis is shown the number of clusters. In bold, best result for every subject.*

Table 4.9 shows the results of different clustering algorithms using VGG-face fine-tuned DCNN model for feature extraction, for every subject in the two different days. As we can see, HAC still gives more best results than the other two algorithms. In this case, rank-order and HAC gives same result in two cases, while k-means outperform them in just one case.

Subject	Rank-Order	HAC	k-Means
Subject1_1	0.108 (93)	**0.148 (8)**	0.145 (2)
Subject1_2	0.066 (103)	0.167 (13)	**0.173 (3)**
Subject2_1	0.139 (5)	**0.143 (4)**	0.134 (5)
Subject2_2	0.145 (20)	**0.192 (2)**	0.134 (8)
Subject3_1	0.123 (59)	**0.222 (20)**	0.220 (2)
Subject3_2	**0.225 (4)**	**0.225 (4)**	**0.225 (4)**
Subject4_1	0.117 (23)	**0.169 (8)**	0.165 (4)
Subject4_2	0.100 (21)	**0.240 (6)**	0.232 (3)

Table 4.9. *Silhouette score for for different clustering algorithms using VGG-face-finetuned DCNN features. In parenthesis is shown the number of clusters. In bold, best result for every subject.*

Comparing the three DCNN features extractors, the best result are given by Openface DCNN in contras to the previous sections (where VGG-face outperformed it). This may be due to how the networks were trained. While Openface were trained using triplet-loss strategy to learn a compact representation of the faces, VGG-face were trained using soft-max strategy, and thus Openface is able to learn a more precise representation of this dataset. As for the fine-tuned version, as explained in section 3.2, it need to be trained using different strategies and perhaps using some sort of data transformation to augment the training data so the network can generalize more.

S1_1	**S1_2**	**S2_1**	**S2_2**	**3_1**	**S3_2**	**S4_1**	**S4_2**
0.202 (2)	0.269 (2)	0.228 (2)	0.309 (2)	0.283 (2)	0.289 (2)	0.225 (2)	0.460 (2)

Table 4.10. *Best Silhouette score for the four subjects. In parenthesis is shown the number of clusters.*

Table 4.10 shown the overall best result for every subject in both days. As we can see, every subject in every day encounters with 2 different group of people (two groups, not the quantity). That means, either every subject only encountered with two different persons or the clustering algorithms found that many different people are similar, so they grouped together in the same cluster. TO check this fact, we need to manually scan the dataset and check the results, but this step is out of the scope of this work and it let to future research lines.

Chapter 5

Conclusions and Future Work

5.1 Summary

In this paper we have presented a system for clustering faces in unconstrained image. We have used and experiment with different deep convolutional neural networks (DCNN) and further fine-tuned our version of VGG-face DCNN using a public dataset. As for clustering phase, we have tested different approaches of clustering algorithms as well as different techniques to align a face before feeding it to a DCNN. The system works as follows, given an input image, (i) localize facial landmarks and extract bounding boxes from the images. With those facial landmarks and the bounding boxes, align the faces such that some points (facial points or calculated ones) remain in the same position among different images). (ii) This aligned face is passed through a DCNN that extract a compact representation of the face, and lastly, but not least, (ii) a clustering algorithm is used to cluster those faces in different groups, so faces similar to each together should fall in the same group and distinct faces should belong to different groups.

Using this system, we achieved results comparable to state of the art, i.e. we achieved 94% of F_1 score on LFW dataset and outperformed state of the art results in some IJB-B clustering protocols (i.e. 94% on IJB-B-128, 94% on IJB-B-256, 85% on IJB-B-512 and 80% on IJB-B-1024 F_1 score).

Further, our objective was to apply this model to egocentric images to study people's social behaviour, which added several more challenges to the list. Egocentric images have less quality, are often blurred, noisy, have occlusions, and their view range is very limited. All these difficulties are specially important in face recognition and clustering problems, as it is necessary to see the whole face or at least an important part of it infer it representation.

5.2 Future Work

Our system relies entirely on third-party trained models, which in one hand are very useful, but limited to the trained datasets (or similar ones) and clustering algorithms. As we saw in the experiments, when clustering images from datasets like, LFW and IJB-B, our system is capable to achieve very high results, but when cluster the EDUB-Obj datasets, the results are not very good. The problem with this kind of datasets is that they are very challenging. Thus, in order to achieve high results, it is a must to fine-tune a pre-existing model, o train a new one from scratch, with this dataset. As we saw, our fine-tuned model were unable to surpass some DCNN we showed in this work. One issue with this is that it need more data to train and tune a lot of parameters. Due to a lack of resources and time, we were unable to tackle this step. In the other hand, using a more sophisticated model, like Res-nets may give good results when clustering face images.

References

[1] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041.

[2] Amos, B., Ludwiczuk, B., and Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science.

[3] Bolaños, M. and Radeva, P. (2015). Ego-object discovery. *arXiv preprint arXiv:1504.01639*.

[4] Cao, Z., Yin, Q., Tang, X., and Sun, J. (2010). Face recognition with learning-based descriptor. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2707–2714. IEEE.

[5] Chen, D., Cao, X., Wen, F., and Sun, J. (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3025–3032.

[6] Chen, J.-C., Patel, V., and Chellappa, R. (2015). Landmark-based fisher vector representation for video-based face verification. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2705–2709. IEEE.

[7] Chen, Y.-C., Patel, V. M., Chellappa, R., and Phillips, P. J. (2014). Adaptive representations for video-based face recognition across pose. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 984–991. IEEE.

[8] Cui, J., Wen, F., Xiao, R., Tian, Y., and Tang, X. (2007). Easyalbum: an interactive photo annotation system based on face clustering and re-ranking. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 367–376. ACM.

[9] Doherty, A. R., Hodges, S. E., King, A. C., Smeaton, A. F., Berry, E., Moulin, C. J., Lindley, S., Kelly, P., and Foster, C. (2013). Wearable cameras in health. *American journal of preventive medicine*, 44(3):320–323.

[10] Gong, Y., Pawlowski, M., Yang, F., Brandy, L., Bourdev, L., and Fergus, R. (2015). Web scale photo hash clustering on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 19–27.

[11] Gowda, K. C. and Krishna, G. (1978). Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition*, 10(2):105–112.

[12] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst.

[13] Huang, Z., Shan, S., Wang, R., Zhang, H., Lao, S., Kuerban, A., and Chen, X. (2015). A benchmark and comparative study of video-based face recognition on cox face database. *IEEE Transactions on Image Processing*, 24(12):5967–5981.

[14] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

[15] Kapoor, A., Hua, G., Akbarzadeh, A., and Baker, S. (2009). Which faces to tag: Adding prior constraints into active learning. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1058–1065. IEEE.

[16] Kazemi, V. and Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874.

[17] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[18] Kurita, T. (1991). An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition*, 24(3):205–209.

[19] Li, S. Z., Lei, Z., and Ao, M. (2009). The hfb face database for heterogeneous face biometrics research. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 1–8. IEEE.

[20] Lin, W.-A., Chen, J.-C., and Chellappa, R. (2017). A proximity-aware hierarchical clustering of faces. *arXiv preprint arXiv:1703.04835*.

[21] Liu, C. and Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image processing*, 11(4):467–476.

[22] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[23] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

[24] Ng, H.-W. and Winkler, S. (2014). A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE.

[25] Otto, C., Jain, A., et al. (2017). Clustering millions of faces by identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[26] Otto, C., Klare, B., and Jain, A. K. (2015). An efficient approach for clustering face images. In *Biometrics (ICB), 2015 International Conference on*, pages 243–250. IEEE.

[27] Parkhi, O. M., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). A compact and discriminative face track descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1693–1700.

[28] Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *British Machine Vision Conference*.

[29] Patel, V. M., Wu, T., Biswas, S., Phillips, P. J., and Chellappa, R. (2012). Dictionary-based face recognition under variable lighting and pose. *IEEE Transactions on Information Forensics and Security*, 7(3):954–965.

[30] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[31] Sankaranarayanan, S., Alavi, A., Castillo, C. D., and Chellappa, R. (2016). Triplet probabilistic embedding for face verification and clustering. In *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on*, pages 1–8. IEEE.

[32] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

[33] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

[34] Shi, Y., Otto, C., and Jain, A. K. (2017). Face clustering: Representation and pairwise constraints. *arXiv preprint arXiv:1706.05067*.

[35] Simonyan, K., Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2013). Fisher vector faces in the wild. In *BMVC*, volume 2, page 4.

[36] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[37] Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996.

[38] Sun, Y., Wang, X., and Tang, X. (2015). Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900.

[39] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708.

[40] Talavera, E., Dimiccoli, M., Bolanos, M., Aghaei, M., and Radeva, P. (2015). R-clustering for egocentric video segmentation. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 327–336. Springer.

[41] Tian, Y., Liu, W., Xiao, R., Wen, F., and Tang, X. (2007). A face annotation framework with partial clustering and interactive labeling. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[42] Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

[43] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

[44] Wang, D., Otto, C., and Jain, A. K. (2015). Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*.

[45] Wang, J., Wang, J., Zeng, G., Tu, Z., Gan, R., and Li, S. (2012). Scalable k-nn graph construction for visual descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1106–1113. IEEE.

[46] Whitelam, C., Taborsky, E., Blanton, A., Maze, B., Adams, J., Miller, T., Kalka, N., Jain, A. K., Duncan, J. A., Allen, K., et al. (2017). Iarpa janus benchmark-b face dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 592–600. IEEE.

[47] Xie, S., Shan, S., Chen, X., and Chen, J. (2010). Fusing local patterns of gabor magnitude and phase for face recognition. *IEEE transactions on image processing*, 19(5):1349–1361.

[48] Yang, J., Parikh, D., and Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156.

[49] Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*.

[50] Zelnik-Manor, L. and Perona, P. (2005). Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608.

[51] Zhang, B., Shan, S., Chen, X., and Gao, W. (2007). Histogram of gabor phase patterns (hgpp): A novel object representation approach for face recognition. *IEEE Transactions on Image Processing*, 16(1):57–68.

[52] Zhao, M., Teo, Y. W., Liu, S., Chua, T.-S., and Jain, R. (2006). Automatic person annotation of family photo album. In *International Conference on Image and Video Retrieval*, pages 163–172. Springer.

[53] Zhu, C., Wen, F., and Sun, J. (2011). A rank-order distance based clustering algorithm for face tagging. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 481–488. IEEE.